Complejidad Computacional.
Máquina de turing sobre alfabeto $b, 0, 1$.

Yani
www.yaniweb.com

8 de febrero de 2014

Resumen

Construcción e implementación de máquina de turing sobre el alfabeto $b, 0, 1$ correspondiente a la asignatura de Complejidad Computacional. Se presenta en este documento el enfoque de solución y manual de usuario. El documento también está en versión PDF.

Contents

1 Problema **1**

2 Solución **2**
2.1 Algoritmo 2
2.2 Especificación formal 2
2.3 Programa 3
2.3.1 Descripción 3
2.3.2 Funciones codificadas 3

3 Manual de Usuario **6**
3.1 Requisitos para ejecución 6
3.2 Ejecución del programa 6
3.3 Uso del programa 7
3.4 Ejemplos Ejecutados 7
3.4.1 Cadena 0011 7
3.4.2 Cadena 000111 8
3.4.3 Cadena 000101 9

1 Problema

Diseñe una máquina de Turing, sobre el alfabeto $b, 0, 1$ (b es el símbolo blanco, y añada, si fuera necesario, símbolos auxiliares), que reconozca al lenguaje consistente de las palabras $0^n 1^n$, con $n \in \mathbb{N}$. Escriba un programa que reciba n y despliegue la manera en la que trabaja la máquina de Turing diseñada.

2 Solución

Se desarrolló un programa que recibe como entrada la cadena en el alfabeto 0^n1^n, por ejemplo admite los siguientes valores: 0011, 000111, 01, 00001111. La idea principal del programa es cambiar primero un 0 por X y luego un 1 por una Y y así sucesivamente hasta que se hayan cambiado todos los ceros y unos.

2.1 Algoritmo

Algorithm 1 Algoritmo de funcionamiento de la MT para leer el alfabeto 0^n1^n

1: Comenzar en extremo izquierdo de la entrada y cambiar el primer 0 por una X. $(0 \leftarrow X)$
2: Mover hacia Izquierda pasando encima de todos los ceros y letras Y hasta encontrar un 1.
3: $1 \leftarrow Y$
4: Mover hacia Izquierda pasando encima de letras Y y ceros hasta encontrar una letra X.
5: Al encontrar la letra X busca un cero inmediatamente a la derecha, y si lo encuentra lo cambia por una X y repite el proceso, cambiando el 1 por una Y.
6: Cuando la entrada no es de la forma 0^n1^n la MT no hace el siguiente movimiento y se para sin aceptar. En caso contrario cuando cambia todos los ceros por X en la misma iteración en que los cambia el último 1 por una Y entonces determina que la entrada es de la forma 0^n1^n y acepta la cadena.

2.2 Especificación formal

Se diseñó una máquina de Turing compuesta de un estado inicial q_0, tres estados posibles y un estado terminal q_4.

La especificación formal de la Máquina de Turing diseñada para este problema es:

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, t, q_0, q_4)$$

La tabla de transiciones t para esta MT se especifica en la figura 3.

	Symbol				
State	0	1	X	Y	B
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	(q_4, B, R)
q_4	–	–	–	–	–

Figure 1: Tabla de transiciones t.

2.3 Programa

2.3.1 Descripción

Se codificó un programa en Java que despliega las Descripciones Instantáneas de la MT y determina si la cadena de entrada está en el alfabeto 0^n1^n. El programa Java está compuesto por las siguientes funciones:

2.3.2 Funciones codificadas

- `main`: Función principal del programa. El programa cuenta con una interfaz visual compuesta de una ventana jFrame de Java.
- `turing_machine`: Esta función contiene el algoritmo encargado de recorrer y leer la cadena de entrada. Recibe como parámetro de entrada la cadena a analizar. Mediante un ciclo while toma el estado actual y el símbolo del cabezal en cada estado e invoca a la función determine_action que contiene las reglas de la tabla de transiciones para determinar el siguiente estado (movimiento de la cinta).
- `determine_action`: Esta función contiene la lógica de reglas de la tabla de transiciones de la figura 3. Recibe como parámetros los siguientes valores:

  1. Estado actual: Compuesto por una cadena numérica: "0", "1", "2", "3", "4"
  2. Símbolo que lee el cabezal de la MT: 0,1,X,Y,B donde X es una variable auxiliar para sustituir por ceros, Y es usada para sustituir por unos, y B representa el espacio en blanco así como también término de la cadena.

  Con los valores anteriores, se consulta en las reglas el estado siguiente, el caracter que debe ser sustituido (0,1,X,Y,B), y el movimiento siguiente Izquierda o Derecha (I, R), los cuales son regresados en un arreglo de 3 elementos donde el elemento de la posición 0 es el estado siguiente, el de la posición 1 el caracter a sustituir, y en la posición 2 el movimiento siguiente.

En el código fuente entregado con este documento, se encuentra el proyecto "MT_alfabeto" (proyecto de tipo Java) en el cual se define el paquete "mx.cinvestav.complexity.MTalfabeto" el cual contiene al archivo MTwindow.java, dentro del cual se pueden encontrar las funciones descritas anteriormente.

Los códigos de estas funciones se presentan a continuación:

Listing 1: Función `determine_action` que representa a la tabla de transiciones t.

```
1  private int determine_action(String state,
2                  String symbol, String[] outputActions){
3          if(state.equals("0") && symbol.equals("0")){
4              outputActions[0]="1";
5              outputActions[1]="X";
6              outputActions[2]="R";
7              return 1; //Symbol satisfy the alfabet
8          }
9          if(state.equals("0") && symbol.equals("Y")){
10             outputActions[0]="3";
```

```
            outputActions[1]="Y";
            outputActions[2]="R";
            return 1; //Symbol satisfy the alfabet
        }
        if(state.equals("1") && symbol.equals("0")){
            outputActions[0]="1";
            outputActions[1]="0";
            outputActions[2]="R";
            return 1; //Symbol satisfy the alfabet
        }
        if(state.equals("1") && symbol.equals("1")){
            outputActions[0]="2";
            outputActions[1]="Y";
            outputActions[2]="L";
            return 1; //Symbol satisfy the alfabet
        }
        if(state.equals("1") && symbol.equals("Y")){
            outputActions[0]="1";
            outputActions[1]="Y";
            outputActions[2]="R";
            return 1; //Symbol satisfy the alfabet
        }
        if(state.equals("2") && symbol.equals("0")){
            outputActions[0]="2";
            outputActions[1]="0";
            outputActions[2]="L";
            return 1; //Symbol satisfy the alfabet
        }
        if(state.equals("2") && symbol.equals("X")){
            outputActions[0]="0";
            outputActions[1]="X";
            outputActions[2]="R";
            return 1; //Symbol satisfy the alfabet
        }
        if(state.equals("2") && symbol.equals("Y")){
            outputActions[0]="2";
            outputActions[1]="Y";
            outputActions[2]="L";
            return 1; //Symbol satisfy the alfabet
        }
        if(state.equals("3") && symbol.equals("Y")){
            outputActions[0]="3";
            outputActions[1]="Y";
            outputActions[2]="R";
            return 1; //Symbol satisfy the alfabet
        }
        if(state.equals("3") && symbol.equals("B")){
            outputActions[0]="4";
            outputActions[1]="B";
            outputActions[2]="R";
```

```
61              return 1; //Symbol satisfy the alfabet
62          }
```

Listing 2: Función turing_machine:.

```
1   private void turing_machine(String word){
2           /*
3           [...]
4           declaracion de variables de interfaz grafica
5           [...]
6           */
7           while(state!="4"){
8               if(this.determine_action(state, symbol, outputActions)==0){
9                   break;
10              }
11              state=outputActions[0];
12              char_symbol = outputActions[1];
13
14              /*Replace the variable*/
15              replaced= new StringBuilder(word);
16              replaced.setCharAt(i, char_symbol.charAt(0));
17              word = replaced.toString();
18
19              /* Add state number for strings of text area only*/
20              replaced2 = replaced;
21              replaced2.insert(i+1, "q{"+state+"}");
22              word2 = replaced2.toString();
23
24              if(outputActions[2].equals("R")){
25                  //Move to Right
26                  i++;
27              } else {
28                  //Move to Left
29                  i--;
30              }
31              this.jtxtOutput.append("\n");
32              this.jtxtOutput.append(word2);
33              /*Validates for final state*/
34              if(state.equals("4")){
35                 this.jlblResultado.setText("CADENA ACEPTADA.");
36                 this.jtxtOutput.append("\n");
37                 this.jtxtOutput.append("CADENA ACEPTADA");
38              }
39
40              if(i<word.length()){
41                  symbol = word.substring(i,i+1);
42              }
43              if(symbol.equals(" ")){
44                  symbol="B";
```

```
45                 }
46                 if(state=="4" && i<word.length()){
47                     state="0";
48                 }
49             }
50         }
```

3 Manual de Usuario

3.1 Requisitos para ejecución

El programa es una aplicación Java sencilla con interfaz gráfica (usando un Java jFrame). Fue construida en Mac OSX usando Netbeans. La forma más fácil para ejecutarla es mediante una línea de comandos invocando al ejecutable como se describe en la siguiente sección. Adicionalmente si se requiere revisar el código fuente y recompilar, se puede abrir el proyecto en el IDE Netbeans.

Requisitos:

1. Cualquier Sistema Operativo con la Java Virtual Machine instalada.

3.2 Ejecución del programa

Pasos para ejecución:

1. Descomprimir el tarball de archivos en la carpeta deseada de su equipo.
2. Abrir una línea de comandos e ir a la carpeta del paso 1
3. Escribir lo siguiente en la línea de comandos y presionar enter:

   java -jar MT_alfabeto/dist/MT_alfabeto.jar

   la imágen 2 muestra la línea de comandos.

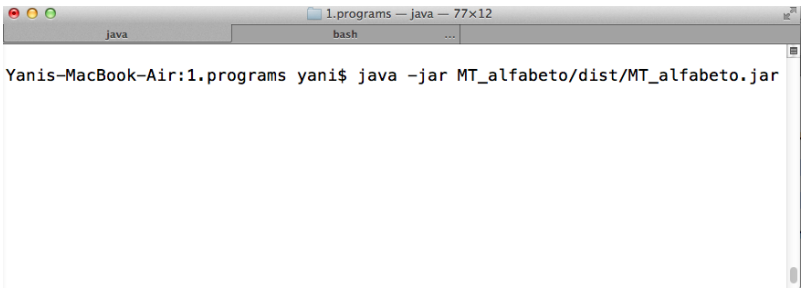

Figure 2: Ejecución de la aplicación en línea de comandos

lo anterior ejecutará la ventana principal de la aplicación la cual se muestra en la figura 3.

Complejidad Computacional - Programa 11
Máquina de Turing de palabras {0^n 1^n}
Cadena (ej: 01, 0011, 000111):
Reconocer
Resultado:
Descripciones instantáneas de la MT:
Notaciones:
q(i): nó. de estado de la MT.
Casilla leída es la casilla adyacente a la derecha de q(i).

Figure 3: Programa en ejecución

3.3 Uso del programa

En la figura 3 se muestra el programa en ejecución. Para ingresar cadenas se utiliza el cuadro de texto a un lado del botón "Reconocer". Una vez ingresada se da clic en reconocer lo cual hace que se desplieguen los movimientos de la MT (descripciones instantáneas) en la área de texto así como el resultado final de la evaluación: Cadena aceptada ó no aceptada.

Los estados se representan en la notación instantánea por $q\{i\}$, y el símbolo que se está leyendo en el cabezal es el adyacente a la derecha de $q\{i\}$.

3.4 Ejemplos Ejecutados

3.4.1 Cadena 0011

La figura 4 muestra el comportamiento de la MT para la cadena 0011 la cual es una cadena correcta en el lenguaje del problema.

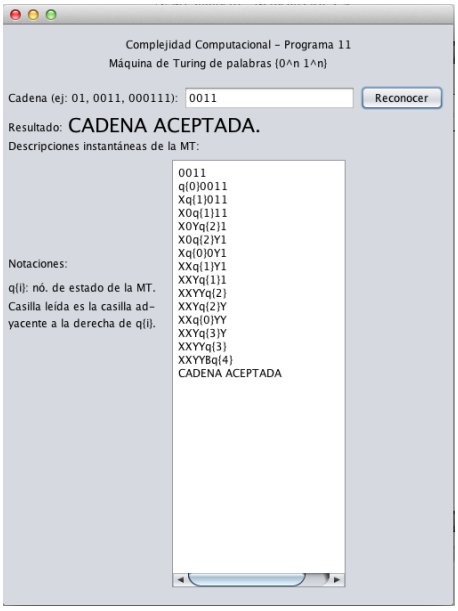

Figure 4: Cadena de ejemplo 0011

3.4.2 Cadena 000111

La figura 5 muestra el comportamiento de la MT para la cadena 000111 la cual es una cadena correcta en el lenguaje del problema.

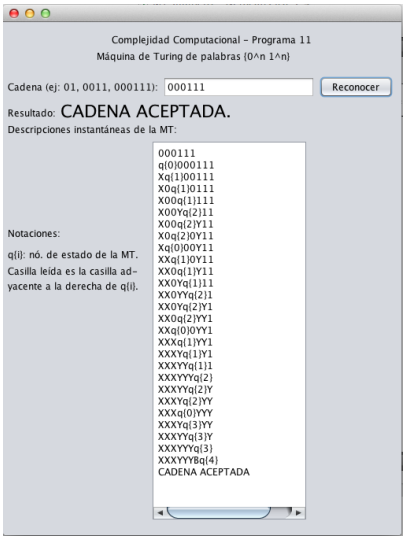

Figure 5: Cadena de ejemplo 000111

3.4.3 Cadena 000101

La figura 6 muestra el comportamiento de la MT para la cadena 000101 la cual **no es** una cadena correcta en el lenguaje del problema.

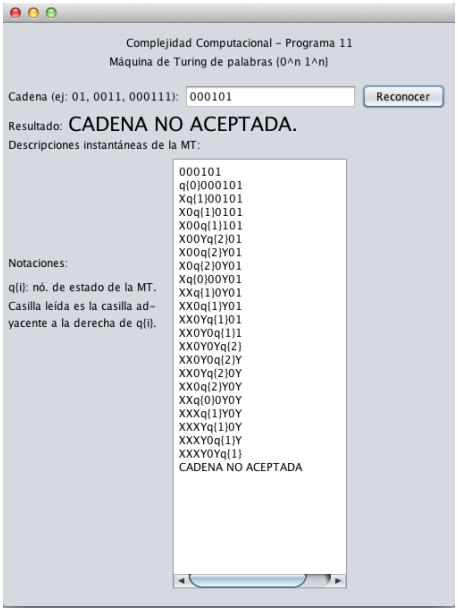

Figure 6: Cadena de ejemplo 000101, la cual no es reconocida por el alfabeto.